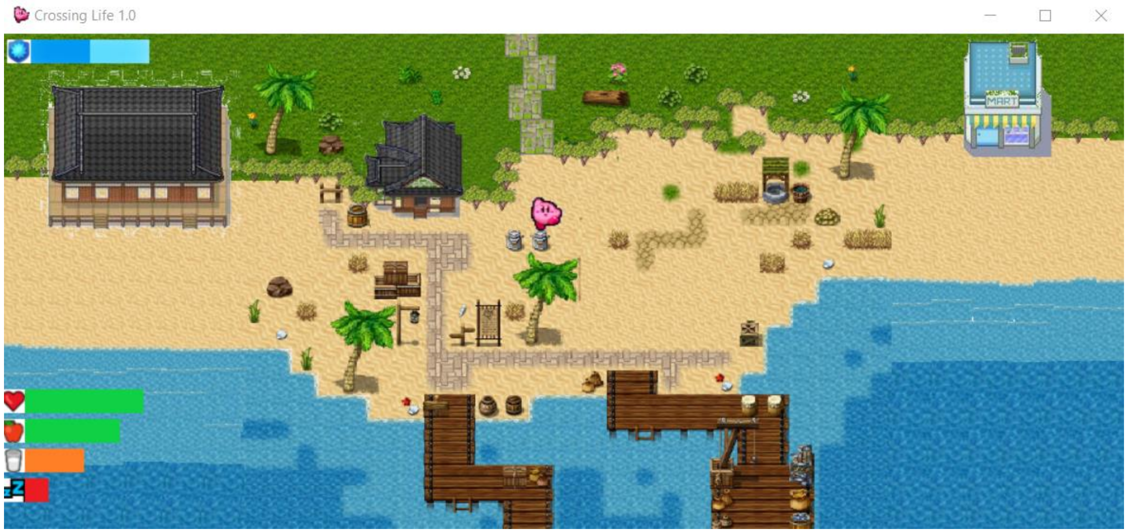
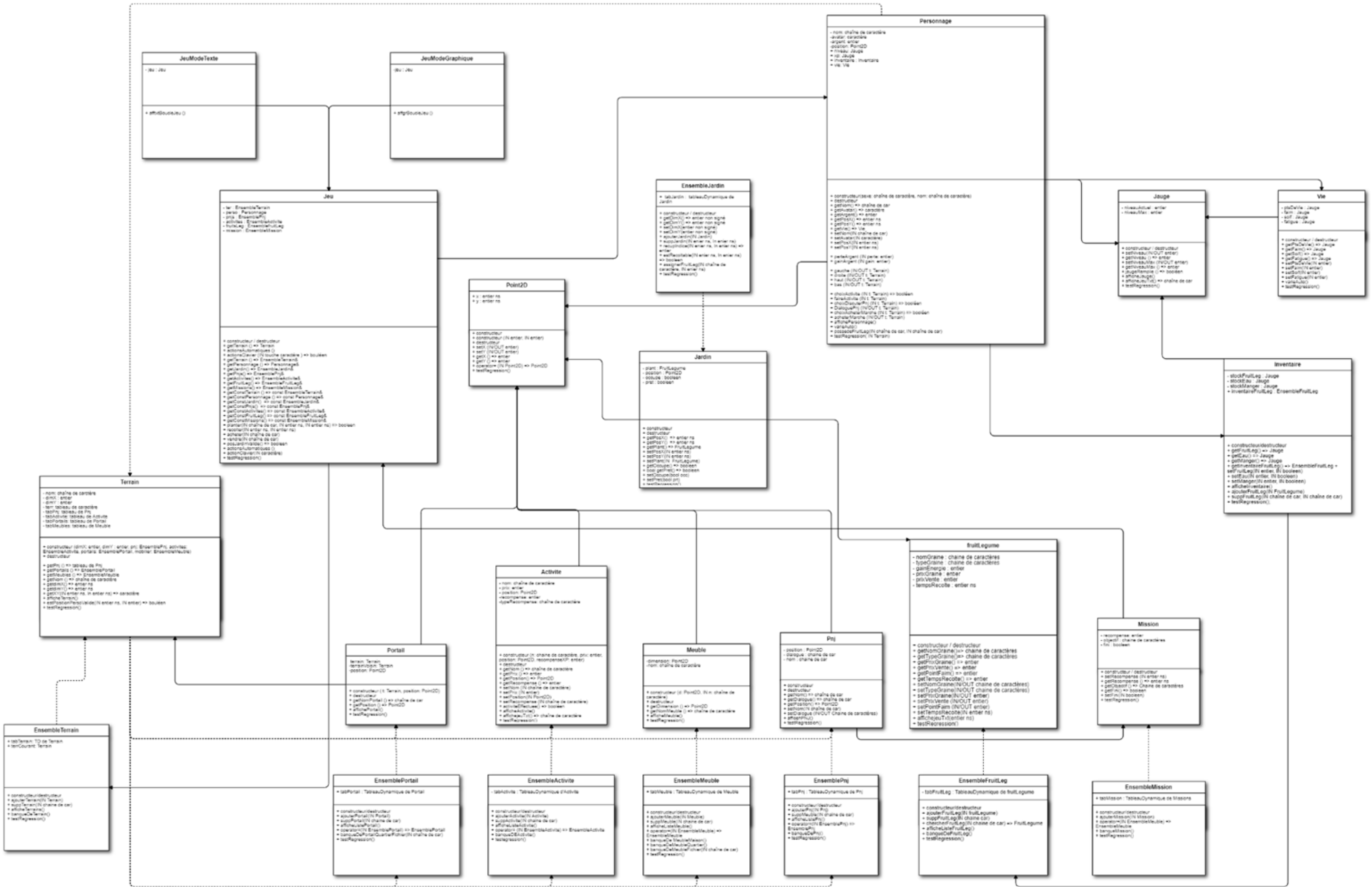
Life Crossing

Simule le concept de vie artificielle, ce jeu est inspiré d’Animal crossing.



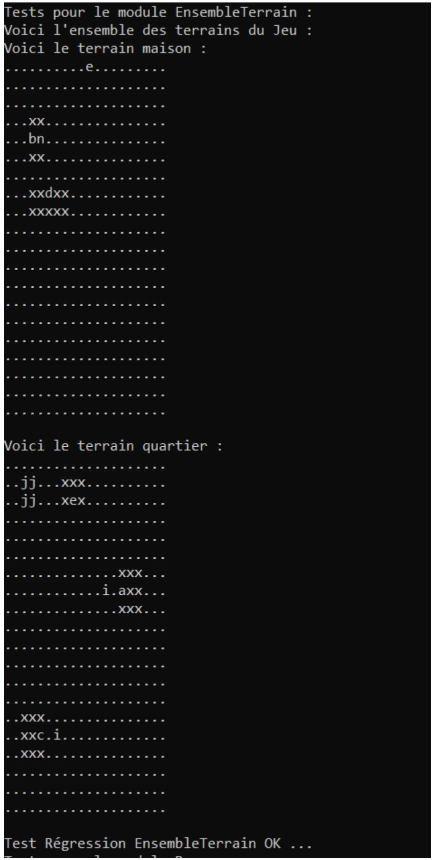

JeuModeTexte
- jeu : Jeu
+ affichBoucleJeu ()
JeuModeGraphique
-jeu : Jeu
+ affgrBoucleJeu ()
Jeu
- ter : EnsembleTerrain
- perso : Personnage
- pnjs : EnsemblePnj
- activites : EnsembleActivite
- fruitsLeg : EnsembleFruitLeg
- mission : EnsembleMission
+ constructeur / destructeur
+ getTerrain () => Terrain
+ actionsAutomatiques ()
+ actionsClavier (IN touche caractère) => booléen
+ getTerrain () => EnsembleTerrain&
+ getPersonnage () => Personnage&
+ getJardin() => EnsembleJardin&
+ getPnjs() => EnsemblePnj&
+ getActivites() => EnsembleActivite&
+ getFruitLeg() => EnsembleFruitLeg&
+ getMissions() => EnsembleMission&
+ getConstTerrain () => const EnsembleTerrain&
+ getConstPersonnage () => const Personnage&
+ getConstJardin() => const EnsembleJardin&
+ getConstPnjs() => const EnsemblePnj&
+ getConstActivites() => const EnsembleActivite&
+ getConstFruitLeg() => const EnsembleFruitLeg&
+ getConstMissions() => const EnsembleMission&
+ planter(IN chaîne de car, IN entier ns, IN entier ns) => boolean
+ recolter(IN entier ns, IN entier ns)
+ acheter(IN chaîne de car)
+ vendre(IN chaîne de car)
+ posJardinValide() => boolean
+ actionsAutomatiques ()
+ actionClavier(IN caractère)
+ testRegression()
Personnage
- nom: chaîne de caractère
-avatar: caractère
-argent: entier
-position: Point2D
+ niveau: Jauge
+ xp: Jauge
+ inventaire : Inventaire
+ vie: Vie
+ constructeur(sexe: chaîne de caractère, nom: chaîne de caractère)
+ destructeur
+ getNom() => chaîne de car
+ getAvatar() => caractère
+ getArgent() => entier
+ getPosX() => entier ns
+ getPosY() => entier ns
+ getVie() => Vie
+ setNom(IN chaîne de car)
+ setAvatar(IN caractère)
+ setPosX(IN entier ns)
+ setPosY(IN entier ns)
+ perteArgent (IN perte: entier)
+ gainArgent (IN gain: entier)
+ gauche (IN/OUT t: Terrain)
+ droite (IN/OUT t: Terrain)
+ haut (IN/OUT t: Terrain)
+ bas (IN/OUT t: Terrain)
+ choixActivite (IN t: Terrain) => booléen
+ faireActivite (IN t: Terrain)
+ choixDiscuterPnj (IN t: Terrain) => booléen
+ DialoguePnj (IN/OUT t: Terrain)
+ choixAcheterMarche (IN t: Terrain) => booléen
+ acheterMarche (IN/OUT t: Terrain)
+ affichePersonnage()
+ varieAuto()
+ possedeFruitLeg(IN chaîne de car, IN chaîne de car)
+ testRegression(IN Terrain)
Jauge
- niveauActuel : entier
- niveauMax : entier
+ constructeur / destructeur
+ setNiveau(IN/OUT entier)
+ getNiveau () => entier
+ setNiveauMax (IN/OUT entier)
+ getNiveauMax () => entier
+ jaugeRemplie () => booléen
+ afficheJauge()
+ afficheJeuTxt() => chaîne de car
+ testRegression()
Vie
- ptsDeVie : Jauge
- faim : Jauge
- soif : Jauge
- fatigue : Jauge
+ constructeur / destructeur
+ getPtsDeVie() => Jauge
+ getFaim() => Jauge
+ getSoif() => Jauge
+ getFatigue() => Jauge
+ setPtsDeVie(IN entier)
+ setFaim(IN entier)
+ setSoif(IN entier)
+ setFatigue(IN entier)
+ varieAuto()
+ testRegression()
EnsembleJardin
+ tabJardin : tableauDynamique de Jardin
+ constructeur / destructeur
+ getDimX() => entier non signé
+ getDimY() => entier non signé
+ setDimX(entier non signé)
+ setDimY(entier non signé)
+ ajouterJardin(IN Jardin)
+ suppJardin(IN enier ns, In enier ns)
+ recupIndice(IN enier ns, In enier ns) => entier
+ estRecoltable(IN enier ns, In enier ns) => booleen
+ assignerFruitLeg(IN chaîne de caractère, IN enier ns)
+ testRegression()
Point2D
+ x : entier ns
+ y : entier ns
+ constructeur
+ constructeur (IN entier, IN entier)
+ destructeur
+ setX (IN/OUT entier)
+ setY (IN/OUT entier)
+ getX () => entier
+ getY () => entier
+ operator+ (IN Point2D) => Point2D
+ testRegression()
Jardin
- plant : FruitLegume
- position : Point2D
- occupe : booleen
- pret : booleen
+ constructeur
+ destructeur
+ getPosX() => entier ns
+ getPosY() => entier ns
+ getPlant() => FruitLegume
+ setPosX(IN entier ns)
+ setPosY(IN entier ns)
+ setPlant(IN FruitLegume)
+ getOccupe() => booleen
+ bool getPret() => booleen
+ setOccupe(bool occ)
+ setPret(bool prt)
+ testRegression()
Inventaire
- stockFruitLeg : Jauge
- stockEau : Jauge
- stockManger : Jauge
+ inventaireFruitLeg : EnsembleFruitLeg
+ constructeur/destructeur
+ getFruitLeg() => Jauge
+ getEau() => Jauge
+ getManger() => Jauge
+ getInventaireFruitLeg() => EnsembleFruitLeg +
setFruitLeg(IN entier, IN booleen)
+ setEau(IN entier, IN booleen)
+ setManger(IN entier, IN booleen)
+ afficheInventaire()
+ ajouterFruitLeg(IN FruitLegume)
+ suppFruitLeg(IN chaîne de car, IN chaîne de car)
+ testRegression();
Terrain
- nom: chaîne de carctère
- dimX : entier
- dimY : entier
- terr: tableau de caractère
- tabPnj: tableau de Pnj
- tabActivite: tableau de Activite
- tabPortails: tableau de Portail
- tabMeubles: tableau de Meuble
+ constructeur (dimX: entier, dimY : entier, pnj: EnsemblePnj, activites: EnsembleActivite, portails: EnsemblePortail, mobilier: EnsembleMeuble)
+ destructeur
+ getPnj () => tableau de Pnj
+ getPortails () => EnsemblePortail
+ getMeubles () => EnsembleMeuble
+ getNom () => chaîne de caractère
+ getdimX() => entier ns
+ getdimY() => entier ns
+ getXY(IN entier ns, In entier ns) => caractère
+ afficheTerrain()
+ estPositionPersoValide(IN entier ns, IN entier) => bouléen
+ testRegression()
Portail
-terrain: Terrain
-terrainVoisin: Terrain
-position: Point2D
+ constructeur (t: Terrain, position: Point2D)
+ destructeur
+ getNomPortail () => chaîne de car
+ getPosition () => Point2D
+ affichePortail()
+ testRegression()
Activite
- nom: chaîne de caractère
- prix: entier
- position: Point2D
-recompense: entier
-typeRecompense: chaîne de caractère
+ constructeur (n: chaîne de caractère, prix: entier, position: Point2D, recompenseXP: entier)
+ destructeur
+ getNom () => chaîne de caractère
+ getPrix () => entier
+ getPosition() => Point2D
+ getRecompense () => entier
+ setNom (IN chaîne de caractère)
+ setPrix (IN entier)
+ setPosition(IN Point2D)
+ setRecompense (IN chaîne de caractère)
+ activiteEffectuee() => booléen
+ afficheActivite()
+ affichejeuTxt() => chaîne de caractère
+ testRegression()
Meuble
-dimension: Point2D
-nom: chaîne de caractère
+ constructeur (d: Point2D, IN n: chaîne de caractère)
+ destructeur
+ getDimension () => Point2D
+ getNomMeuble () => chaîne de caractère
+ afficheMeuble()
+ testRegression()
Pnj
- position : Point2D
- dialogue : chaîne de car
- nom : chaîne de car
+ constructeur
+ destructeur
+ getNom() => chaîne de car
+ getDialogue() => chaîne de car
+ getPosition() => Point2D
+ setNom(IN chaîne de car)
+ setDialogue (IN/OUT Chaîne de caractères)
+ affichePNJ()
+ testRegression()
fruitLegume
- nomGraine : chaine de caractères
- typeGraine : chaine de caractères
- gainEnergie : entier
- prixGraine : entier
- prixVente : entier
- tempsRecolte : entier ns
+ constructeur / destructeur
+ getNomGraine()=> chaine de caractères
+ getTypeGraine()=> chaine de caractères
+ getPrixGraine() => entier
+ getPrixVente() => entier
+ getPointFaim() => entier
+ getTempsRecolte() => entier
+ setNomGraine(IN/OUT chaine de caractères)
+ setTypeGraine(IN/OUT chaine de caractères)
+ setPrixGraine(IN/OUT entier)
+ setPrixVente (IN/OUT entier)
+ setPointFaim (IN/OUT entier)
+ setTempsRecolte(IN entier ns)
+ affichejeuTxt(entier ns)
+ testRecression()
Mission
- recompense: entier
- objectif : chaine de caractères
- fini : booleen
+ constructeur / destructeur
+ setRecompense (IN entier ns)
+ getRecompense () => entier ns
+ getObjectif () => Chaine de caractères
+ getFini() => booleen
+ setFini(IN booleen)
+ testRegression()
EnsembleTerrain
+ tabTerrain: TD de Terrain
+ terrCourant: Terrain
+ constructeur/destructeur
+ ajouterTerrain(IN Terrain)
+ suppTerrain(IN chaine de car)
+ afficheTerrains()
+ banqueDeTerrain()
+ testRegression()
EnsemblePortail
+ tabPortail : TableauDynamique de Portail
+ constructeur/destructeur
+ ajouterPortail(IN Portail)
+ suppPortail(IN chaine de car)
+ afficheListePortail()
+ operator+(IN EnsemblePortail) => EnsemblePortail
+ banqueDePortailQuartierFichier(IN chaîne de car)
+ testRegression()
EnsembleActivite
- tabActivite : TableauDynamique d'Activite
+ constructeur/destructeur
+ ajouterActivite(IN Activite)
+ suppActivite(IN chaine de car)
+ afficheListeActivite()
+ operator+ (IN EnsembleActivite) => EnsembleActivite
+ banqueDEActivite()
+ testegression()
EnsembleMeuble
+ tabMeuble : TableauDynamique de Meuble
+ constructeur/destructeur
+ ajouterMeuble(IN Meuble)
+ suppMeuble(IN chaine de car)
+ afficheListeMeuble()
+ operator+(IN EnsembleMeuble) => EnsembleMeuble
+ banqueDe MeubleMaison()
+ banqueDeMeubleQuartier()
+ banqueDeMeubleFichier(IN chaîne de car)
+ testRegression()
EnsemblePnj
+ tabPnj : TableauDynamique de Pnj
+ constructeur/destructeur
+ ajouterPnj(IN Pnj)
+ suppMeuble(IN chaine de car)
+ afficheListePnj()
+ operator+(IN EnsemblePnj) => EnsemblePnj
+ banqueDePnj()
+ testRegression()
EnsembleFruitLeg
- tabFruitLeg : TableauDynamique de fruitLegume
+ constructeur/destructeur
+ ajouterFruitLeg(IN fruitLegume)
+ suppFruitLeg(IN chaine car)
+ chercherFruitLeg(IN chaine de car) => FruitLegume
+ afficheListeFruitLeg()
+ banqueDeFruitLeg()
+ testRegression()
EnsembleMission
+ tabMission : TableauDynamique de Missions
+ constructeur/destructeur
+ ajouterMission(IN Mission)
+ operator+(IN EnsembleMeuble) => EnsembleMeuble
+ banqueMission()
+ testRegression()

EnsembleTerrain.h

Ce module permet de créer et relier tous les terrains et de définir le terrain où le personnage se trouve en temps réel.

Données membre :
vector<Terrain> tabTerrain : Tableau Dynamique contenant tous les terrains du jeu.
Terrain terrCourant : Terrain où se trouve le personnage.

Fonctions membres :
EnsembleTerrain() : crée un tableau vide.
~EnsembleTerrain() : Vide le tableau.
affciheTerrain() : parcous tous le tableau et affiche chaque terrains.
void banqueDeTerrain() : crée un ensemble de portail et un ensemble de meubles pour chaque terrains et ajoute des caractères pour les éléments avec peu d'instances.
void testRegression() : test toutes le fonctions du module avec des assert.



```
Tests pour le module EnsembleTerrain :
Voici l'ensemble des terrains du Jeu :
Voici le terrain maison :
..........e.........
....................
....................
...xx...............
...bn...............
...xx...............
....................
...xxdxx............
...xxxxx............
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................

Voici le terrain quartier :
....................
..jj...xxx..........
..jj...xex..........
....................
....................
....................
..............xxx...
............i.axx...
..............xxx...
....................
....................
....................
....................
....................
..xxx...............
..xxc.i.............
..xxx...............
....................
....................
....................

Test Régression EnsembleTerrain OK ...
```

Personnage.h

Ce module permet de gérer les différentes caractéristiques du personnes.

Données membre :
String nom : nom du personnage

char avatar : l’avatar du personnage

int argent : l’argent du personnage

Point2D position : la position du personnage

Vie vie : l’argent du personnage

Jauge niveau : le niveau

Jauge xp : les points d’xp du personnage

Inventaire inventaire : l’inventaire du personnage

Fonctions membres :
Personnage() : crée un personnage avec des données initiales
Mutateurs/seteurs :sert à accéder et modifier les données passées en privé

bas(Terrain &t), **haut(Terrain &t), droite(Terrain &t), gauche(Terrain &t)**:sert à modifier la position du personnage dans le terrain passé en argument

void affichePersonnage() : affiche les données du personnage

void varieAuto() : effectue les changements des données de niveaux et d’xp en fonction de l’avancement de la partie
void testRegression() : test toutes le fonctions du module avec des assert.

sdlJeu.h

Ce module fait le lien entre tous les modules implémentés et les images du jeux.

void jouer(SDL_Surface * ecran,Jeu & jeu,Map & map) : gère toutes les actions du terrain principal, coeur du jeu version graphique
void teleporter(SDL_Surface * ecran,int nb_carte,Jeu & jeu,Map & map) : gère le changement de carte et actions réalisées en dehors du terrain principal
void deplacerJoueur(SDL_Rect *pos,int direction,SDL_Surface *ecran,int & n,int nb_carte,Jeu & jeu,Map & map) : gère les déplacements du joueur et les collisions
void sdlPlanter(SDL_Surface *ecran,const SDL_Rect *pos,bool &b,Jeu & jeu,Map & map) : gère l'interface jardin

string transformeConstantes(int n) : effectue la conversion d'un int en string
void dialoguePNJ(SDL_Surface * ecran,SDL_Rect *pos,Map & map) : gère les dialogues avec les PNJ
void affichageJauge(SDL_Surface * ecran, Jeu & jeu) : affiche les jauges
void affichageMission(SDL_Surface * ecran, Jeu & jeu) : affiche la mission en cours
void affichageReussie(SDL_Surface * ecran, Jeu & jeu, int miss) : affiche un message en cas de succès d'une mission
void manger(Jeu & jeu) : fait manger le personnage
void dormir(Jeu & jeu) : fait dormir le personnage
void boire(Jeu & jeu) : fait boire le personnage

Conclusion

Réalisations : Déplacement, vie (dormir, manger et boire), remporter et perdre de l'argent, planter, récolter,dialoguer, effectuer des missions, acheter de l'eau et des graines, et vendre nos fruits et légumes.

Non réalisé : Affichage de texte sur la fenêtre (pour afficher l'argent ou le niveau du personnage par exemple), on a cependant contrer cela pour les dialogues en affichant des images de dialogue. Système d'avatar pour changer l'apparence. Création du jardin dans la version texte ainsi que la vente des fruits légumes, pas le temps en version texte.

Avec plus de temps : Nouveaux fruits et légumes disponibles selon le niveau par exemple. Implémentation de plus d'activités, missions, … création de minis jeux interactifs en guise de missions(défi) et sauvegarde de la partie. Système de sauvegarde. Davantage de missions

Difficultés : Gestion de la variation des jauges (pas lié au temps mais aux nombres de pas). Pas de temps de pousse pour les fruits/légumes. En générale, la création d'une fonction qui dépend du temps.

Respect du cahier des charges : Un peu de retard sur la version txt, + de temps que prévu. Délais d'implémentation des modules et version graphique respectés. Utilisation de la lib SDL1 au lieu de SDL2.